Remarks

Entry of the amendments, reconsideration of the application, as amended, and allowance of all pending claims are respectfully requested. Claims 1-22, 24 and 27-44 are pending.

Applicants have amended the originally filed independent claims to further define the environment of one aspect of the invention. For example, the originally filed independent claims have been amended to indicate that the program which is being debugged does not include embedded debug commands. Support for this amendment is provided throughout applicants' specification. For example, the exemplary source code depicted on pages 20 and 21 are devoid of debug commands. That is, applicants' specification and the examples provided therein depict source code with no debug commands, providing support that the program does not include embedded debug commands. Thus, no new matter is added.

Moreover, applicants have canceled claims 23, 25 and 26, without prejudice, and have included a new independent claim, claim 42, and dependent claims 43-44 to further define an aspect of applicants' invention. Support for these claims can be found throughout the specification (e.g., paragraph 61, p. 20; paragraphs 64-70, pages 21-22; FIG. 2, FIGs. 5A-5B). Thus, no new matter is added.

In the Office Action, dated September 2, 2004, claims 1-11, 13-24, 26-39 and 41 are rejected under 35 U.S.C. 102(b) as being anticipated by U.S. Patent No. 5,815,714 to Shridhar et al. Additionally, claims 12, 25 and 40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Shridhar in view of U.S. Patent No. 5,819,093 to Davidson et al. Applicants respectfully, but most strenuously, traverse these rejections for the reasons below.

Applicants' invention is directed, in one example, to automatically restoring debugging breakpoints subsequent to program code modification. The program being debugged does not include embedded debug commands. Instead, breakpoints are provided by a debugger during a debug session.

In order to automatically restore a debugging breakpoint to the same step of the program that had the breakpoint prior to modification, the location (e.g., line of code) that

includes that selective step needs to be determined. This determination is made by creating an instruction profile that includes attributes of a number of generated instructions that are associated with the selected step. These attributes include attributes of the instruction (such as, operands and operation type obtained from reading the instruction) of the selected step, and possibly attributes of other instructions in proximity to the selected step. This is described further with reference to the following program:

Source View On Debugger Front End	**Machine Instructions In Application**
Line 1: /*Test program 1*/	00001 L 1,4,0
Line 2: int i = 0;	00002 L 1,5,1
Line 3: int j = 1;	00003 TC 1,4,5
Line 4: if (i > j) {	00004 BL 6
Line 5: print hello;	00005 P hello
Line 6: }	00006 TC 1,4,5
Line 7: if (j > i) {	00007 BH 9
Line 8: print world;	00008 P world
Line 9: }	00009 G back
Line 10: exit;	

In this example, a line breakpoint is set for line 7 in the Source View, which corresponds to line 6 of the Machine Instructions. However, to describe line 6, which looks like line 3, attributes of line 7 of the Machine Instructions are also included in the instruction profile in order to uniquely identify line 6 as the line in which the breakpoint is associated with. The number of instructions to be included in the instruction profile is determined based on a calibration technique described in applicants' specification.

When the program is modified, the attributes of the instruction profile are compared to the attributes of the newly generated instructions to determine where the particular line of code of interest (e.g., line 6) has moved in the program. The various comparisons made to different instructions within the modified program yield difference counters. In this example, the difference counter having the smallest value indicates the location of the selected step. Thus, the debugger automatically restores the breakpoint to that step.

In one particular example, applicants claim a method of restoring debugging breakpoints, in which the method includes, for instance, having a breakpoint that is set to a

selected step of a program, the program being absent of embedded debug commands; and automatically restoring, after modification of the program, the breakpoint to the selected step, wherein the selected step is at a different location within the modified program. Thus, in this aspect of applicants' claimed invention, debugging breakpoints are automatically restored in an environment in which the program being debugged does not include embedded debug commands. This is in sharp contrast to Shridhar.

In Shridhar, it is explicitly stated in many places that the program is to include embedded debug commands. For example, FIG. 1B explicitly depicts embedded debug commands, as well as FIGs. 2 and 3. Further, the manner in which the embedded debug commands are processed is depicted in FIGs. 4-5. Moreover, in Shridhar, the text explicitly describes the generation of source code having embedded debug commands (e.g., Col. 5, lines 51-55; Col. 6, lines 12-15), the extraction of embedded debug commands (e.g., Col. 6, lines 12-38), and the processing of those extracted embedded debug commands (e.g., Col. 6, lines 39-67). The inclusion of debug commands in the source code is imperative for the process of Shridhar. Without the debug commands in the source code, the Shridhar process would not work.

In contrast, applicants have determined a way to automatically restore breakpoints without requiring debug commands to be embedded in the source code. In applicants' invention, a capability is described that enables breakpoint locations to be determined without requiring those breakpoints to be included in the source code. This is very different from Shridhar that explicitly requires embedded debug commands in the source code.

There is absolutely no teaching in Shridhar of how to automatically restore breakpoints when no debug commands are in the source code. As a matter of fact, Shridhar teaches away from this by indicating that having no debug commands in the source code is the prior art (see, e.g., FIG. 1a) and that they are providing something different. One skilled in the art looking at the teachings of Shridhar would understand that in order to automatically restore breakpoints, embedded debug commands would need to be placed in the source code. There is no description, teaching or suggestion of how this would be accomplished without embedded debug commands. Thus, Shridhar does not anticipate applicants' claimed

invention in which a capability is provided for automatically restoring breakpoints in an environment in which programs do not require embedded debug commands.

Based on the foregoing, applicants respectfully submit that their invention is not described, taught or suggested by Shridhar. Therefore, applicants respectfully request an indication of allowability for independent claim 1, any similar independent claims, and all claims depending therefrom.

As a further example, applicants claim a method of restoring debugging breakpoints, in which the method includes, for instance, having a breakpoint that is set to a selected step of a program; and automatically restoring, after modification of the program, the breakpoint to the selected step, wherein the selected step is at a different location within the modified program. The automatically restoring includes, for instance, creating an instruction profile that includes one or more attributes of one or more instructions generated for the selected step of the program prior to modification and zero or more attributes of zero or more other instructions to facilitate uniquely identifying a location of the selected step; and comparing one or more attributes of the instruction profile to one or more attributes of the one or more instructions generated for the modified program to determine the location of the selected step to restore the breakpoint to the determined location. Thus, in this aspect of applicants' invention, the location of the selected step in which the breakpoint is to be restored is determined by employing an instruction profile that was generated from the program prior to modification. This instruction profile includes attributes of one or more instructions that were generated for the selected step of the program prior to modification and may include attributes of other instructions that help to uniquely identify the location of that selected step prior to modification. Attributes of the instruction profile are then compared to instructions generated for the modified program to determine the location of the selected step. Once this location is determined, the breakpoint is set to that selected step. This is very different from the teachings of Shridhar.

In Shridhar, debug commands (i.e., breakpoints) are embedded within the source code. This is true of the program prior to modification, as well as of the program after modification. Since the debug commands are embedded within the program, there is no need

to determine the location of the debug commands. Since the debug commands are already at the location in which they are to be set, there is no need to determine the location.

In contrast, in applicants' claimed invention, applicants describe a technique for determining the location at which a breakpoint is to be automatically set after a program is modified. This technique includes, for instance, creating a profile that includes attributes regarding the location of the selected step prior to modification. These attributes are then used to determine where in the program that selected step is now located. No such steps are required in Shridhar, since Shridhar simply embeds the debug commands within the program itself.

Since the environment of Shridhar is very different from the environment of applicants' invention, there is no need in Shridhar, and there is no description, teaching or suggestion in Shridhar, of creating an instruction profile that includes one or more attributes of one or more instructions generated for the selected step of the program prior to modification and zero or more attributes of zero or more other instructions to facilitate uniquely identifying a location of the selected step, nor of comparing one or more attributes of the instruction profile to one or more attributes of one or more instructions generated for the modified program to determine the location of the selected step to restore the breakpoint to the determined location. This is simply missing from Shridhar.

Based on the foregoing, applicants respectfully request an indication of allowability for new independent claim 42. Further, dependent claims 43 and 44 are allowable for the same reasons as the independent claim from which they depend, as well as for their own additional features.

Applicants respectfully submit that all pending claims are patentable over Shridhar, either alone or in combination with Davidson, and therefore, respectfully request an indication of allowability for all pending claims.

Should the Examiner wish to discuss this case with applicants' attorney, please contact applicants' attorney at the below listed number.

Respectfully submitted,

Blanche E. Schiller

Blanche E. Schiller
Attorney for Applicants
Registration No.: 35,670

Dated: December 14, 2004.

HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, New York 12203-5160
Telephone: (518) 452-5600
Facsimile: (518) 452-5579